P1 1189178

REC'D 0 8 JUL 2004
WIPO PCT

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

July 01, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE.

APPLICATION NUMBER: *60/458,886*
FILING DATE: *March 29, 2003*
RELATED PCT APPLICATION NUMBER: *PCT/US04/09645*

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

E. Bornett

E. BORNETT
Certifying Officer

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

03-31-03

PROVISIONAL APPLICATION COVER SHEET [37 CFR 1.53(c)]

This is a request for filing a PROVISIONAL APPLICATION under 35 U.S.C. §111(b) and 37 CFR 1.51(a)(2)

Date : March 28, 2003
Docket No. : 50089/FLC/R268
EXPRESS MAIL NO. EL 904756191 US

Mail to: BOX PROVISIONAL PATENT APPLICATION

INVENTOR(S)/APPLICANT(S) (LAST NAME, FIRST NAME, MIDDLE INITIAL, RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)

Gerla, Dr. Mario, Los Angeles, California

__X__ Additional inventors are being named on separately numbered sheets attached hereto.

TITLE OF THE INVENTION (280 characters max)

METHOD AND APPARATUS FOR EFFECTIVE START UP OF TCP CONNECTIONS OVER HIGH SPEED, LONG PROPAGATION PATHS

APPLICANT(S) STATUS UNDER 37 CFR § 1.27

__X__ Applicant(s) and any others associated with it/them under § 1.27(a) are a SMALL ENTITY

ENCLOSED APPLICATION PARTS

__10__ Specification (number of pages)
__1__ Drawings (number of sheets)
_____ Assignment
__X__ Other (specify): Appendices A, B, C and D

FEE AND METHOD OF PAYMENT

__X__ A check for the filing fee of $__80.00__ is enclosed.
The Commissioner is hereby authorized to charge any fees under 37 CFR 1.16 and 1.17 which may be required by this filing to Deposit Account No. 03-1728. Please show our docket number with any charge or credit to our Deposit Account. **A copy of this letter is enclosed.**

_____ No filing fee enclosed.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

__X__ No ___ Yes, the name of the U.S. Government agency and the Government contract number are:

Please address all correspondence to **CHRISTIE, PARKER & HALE, LLP, P.O. Box 7068, Pasadena, CA 91109-7068, U.S.A.**

Respectfully submitted,
CHRISTIE, PARKER & HALE, LLP

By _____
Frank L. Cire
Reg. No. 42,419
626/795-9900

PROVISIONAL APPLICATION FILING ONLY

FLC/mas
MAS PAS496650.1-*-3/28/03 11:25 PM

PROVISIONAL APPLICATION COVER SHEET
Docket No. 50089/FLC/R268

ADDITIONAL INVENTORS
(LAST NAME, FIRST NAME, MIDDLE INITIAL, RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)

Sanadidi, Dr. M.Y., Los Angeles, California
Pau, Dr. Giovanni, Los Angeles, California
Wang, Ren, Los Angeles, California

50089/FLC/R268

METHOD AND APPARATUS FOR EFFECTIVE START UP OF TCP CONNECTIONS OVER HIGH SPEED, LONG PROPAGATION PATHS

BACKGROUND OF THE INVENTION

TCP is a reliable data transfer protocol used widely over the Internet for numerous applications, from FTP to HTTP. The current implementation of TCP Reno/NewReno mainly includes two phases: Slow-start and Congestion-avoidance. In the Slow-start phase, a sender opens the congestion window (cwnd) exponentially, doubling cwnd every Round-Trip Time (RTT) until it reaches the Slow-start Threshold (ssthresh). The connection switches then to Congestion-avoidance, where cwnd grows more conservatively, by only 1 packet every RTT (or linearly). The initial ssthresh is set to an arbitrary default value, ranging from 4K to 64K Bytes, depending on the operating system implementation.

By setting the initial ssthresh to an arbitrary value, TCP performance may suffer from two potential problems: (a) if ssthresh is set too high relative to the network Bandwidth Delay Product (BDP), the exponential increase of cwnd generates too many packets too fast, causing multiple losses at the bottleneck router and coarse timeouts, with significant reduction of the connection throughput; (b) if the initial ssthresh is set low relative to BDP, the connection exits Slow-start and switches to linear cwnd increase prematurely, resulting in poor startup utilization especially when BDP is large.

Recent studies reveal that a majority of the TCP connections are short-lived (mice), while a smaller number of long-lived connections carry most Internet traffic (elephants). A short-lived connection usually terminates even before it reaches "steady state". That is, before cwnd grows to make good utilization of the path bandwidth. Thus, the startup stage can significantly affect the performance of the mice. In a large BDP network, with the current Slow-start scheme, it takes many RTTs

for a TCP connection to reach the ideal window (equal to BDP). For example, in current Reno/NewReno implementation with initial ssthresh set to 32 Kbytes, a TCP connection takes about 100 sec to reach the ideal window over a path with a bottleneck bandwidth of 100 Mbps and RTT of 100ms. The utilization in the first 10 sec is a meager 5.97%. With the rapid development of the Internet and ever-growing BDP, a more efficient Slow-start mechanism is required to achieve good link-utilization.

A variety of methods have been suggested to avoid multiple losses and achieve higher utilization during the startup phase. A larger initial cwnd, roughly 4K bytes, has been proposed. This could greatly speed up transfers with only a few packets. However, the improvement is still inadequate when BDP is very large, and the file to transfer is bigger than just a few packets. Fast start uses cached cwnd and ssthresh in recent connections to reduce the transfer latency. The cached parameters may be too aggressive or too conservative when network conditions change.

Smooth start [WXRS] has been proposed to slow down cwnd increase when it is close to ssthresh. The assumption here is that default value of ssthresh is often larger than the BDP, which is no longer true in large bandwidth delay networks. In one proposed solution, the initial ssthresh is set to the BDP estimated using packet pair measurements. This method can be too aggressive. In another proposed method, termed Shared Passive Network Discovery (SPAND), has been proposed to derive optimal TCP initial parameters. SPAND needs leaky bucket pacing for outgoing packets, which can be costly and problematic in practice.

TCP Vegas detects congestion by comparing the achieved throughput over a cycle of length equal to RTT, to the expected throughput implied by cwnd and baseRTT (minimum RTT) at the beginning of a cycle. This method is applied in both Slow-start

and Congestion-avoidance phases. During Slow-start phase, a Vegas sender doubles its cwnd only every other RTT, in contrast with Reno's doubling every RTT. A Vegas connection exits slow-start when the difference between achieved and expected throughput exceeds a certain threshold. However, Vegas may not be able to achieve high utilization in large bandwidth delay networks because of its over-estimation of RTT.

SUMMARY

In one aspect of the invention, a method, herein termed Adaptive Start (Astart) is used at start up, or after a timeout occurs. In Astart, when a connection initially begins or re-starts after a coarse timeout, Astart adaptively and repeatedly resets the TCP Slow-start Threshold (ssthresh) based on the Eligible Rate Estimation (ERE), as calculated in TCP Westwood (TCPW). Using ERE provides the means for adapting to network conditions during the startup phase. Thus a sender is able to grow the congestion window (cwnd) quickly without incurring risk of buffer overflow and multiple losses. AStart can significantly improve link utilization under various bandwidth, buffer size and round-trip propagation times. Most importantly, the method avoids both link under-utilization due to premature Slow-start termination, as well as multiple losses due to initially setting ssthresh too high, or increasing cwnd faster than appropriate.

In TCPW, a sender calculates ERE according to our previously disclosed inventions (BE, CRB, or ABSE), and then uses ERE during the congestion avoidance phase of TCP as follows:

```
if (3 DUPACKS are received)
    ssthresh = (ERE*RTTmin)/seg_size;
    if (cwnd >ssthresh)  /*congestion avoid*/
        cwnd=ssthresh;
```

```
        endif
    endif
    if (coarse timeout expires)
        cwnd = 1;
        ssthresh =(ERE *RTTmin)/seg_size;
        if(ssthresh < 2)
            ssthresh = 2;
        endif
    endif
```

In Astart, a sender calculates ERE and uses ERE during start-up or after a Timeout as follows:

```
    if ( 3 DUPACKS are received)
        switch to congestion avoidance phase;
    else (ACK is received)
        if (ssthresh < (ERE*RTTmin)/seg_size)
                ssthresh =(ERE*RTTmin)/seg_size;
        endif
        if (cwnd >ssthresh)  /*mini congestion avoid. phase*/
            increase cwnd by 1/RTT;
        else if cwnd <ssthresh)  /*mini slow start phase*/
            increase cwnd by 1;
        endif
    endif
```

This mode of operation can be extended to the entire lifetime of the connection, thus protecting also against random errors and sudden increases of bottleneck bandwidth, as may occur with nomadic users.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become better understood with regard to the following description, accompanying drawings, and attached appendices where:

FIG. 1 is a block diagram of a computing device suitable for hosting a TCP process in accordance with an exemplary embodiment of the present invention;

Appendix A is a publication describing a first method of calculating an ERE in accordance with an exemplary embodiment of the present invention;

Appendix B is a publication describing a second method of calculating an ERE in accordance with an exemplary embodiment of the present invention;

Appendix C is a publication describing a third method of calculating an ERE in accordance with an exemplary embodiment of the present invention; and

Appendix D is an abbreviated presentation of the present invention.

DETAILED DESCRIPTION

In TCP Westwood (TCPW), the sender continuously monitors ACKs from the receiver and computes its current Eligible Rate Estimate (ERE). ERE relies on an adaptive estimation technique applied to ACK stream. The goal of ERE is to estimate the eligible sending rate for a connection, and thus achieving high utilization without starving other connections. Research on active network estimation reveals that samples obtained using packet pair often reflects physical bandwidth, while samples obtained using a long packet train gives short-time throughput estimates. Not having the luxury to estimate using active probing packets, a TCPW sender carefully chooses sampling intervals and filtering techniques to estimate the eligible bandwidth share of

a connection. DUPACKs and delayed ACKs are also properly counted in ERE computation.

In current TCPW implementation, upon packet loss (indicated by 3 DUPACKs or a timeout) the sender sets cwnd and ssthresh based on the current ERE. TCPW uses the following algorithm to set cwnd and ssthresh.

```
if (3 DUPACKS are received)
    ssthresh = (ERE*RTTmin)/seg_size;
    if (cwnd >ssthresh)  /*congestion avoid*/
        cwnd=ssthresh;
    endif
endif
if (coarse timeout expires)
    cwnd = 1;
    ssthresh =(ERE *RTTmin)/seg_size;
    if(ssthresh < 2)
        ssthresh = 2;
    endif
endif
```

Adaptive Start (Astart), improves TCP startup performance. AStart takes advantage of the Eligible Rate Estimation (ERE) mechanism proposed in TCPW and adaptively and repeatedly resets ssthresh during the slow-start phase. When ERE indicates that there is more available capacity, the connection opens its cwnd faster, enduring better utilization. On the other hand, when ERE indicates that the connection is close to steady state, it switches to Congestion-avoidance, limiting the risk of buffer overflow and multiple losses. As such, AStart significantly enhances performance of TCP connections, and enhancement increases as BDP increases. When BDP reaches around 750 packets, the throughput improvement is an order of magnitude higher than that of TCP Reno/NewReno for short-lived connections.

AStart is a sender-side only modificationto the traditional Reno/NewReno slow start algorithm. The TCPW eligible rate estimate is used to adaptively and repeatedly reset ssthresh during the startup phase, both connection startup, and after every coarse timeout. The pseudo code of the algorithm is as follows. When an ACK arrives:

```
if ( 3 DUPACKS are received)
    switch to congestion avoidance phase;
else (ACK is received)
    if (ssthresh < (ERE*RTTmin)/seg_size)
            ssthresh =(ERE*RTTmin)/seg_size;
    endif
    if (cwnd >ssthresh)  /*mini congestion avoid. phase*/
        increase cwnd by 1/RTT;
    else if cwnd <ssthresh)  /*mini slow start phase*/
        increase cwnd by 1;
    endif
endif
```

In TCPW, an eligible rate estimate is determined after every ACK reception. In Astart, when the current ssthresh is much lower than ERE, the sender resets ssthresh higher accordingly, and increases cwnd in slow-start fashion. Otherwise, cwnd increases linearly to avoid overflow. In this way, Astart probes the available network bandwidth for this connection, and allows the connection to eventually exit Slow-start close to the ideal window. Compared to Vegas, TCPW avoids premature exit of slow start since it relies on both RTT and ACK intervals, while Vegas only relies on RTT estimates.

By applying Astart, the sender does not overflow the bottleneck buffer and thus multiple losses are avoided. In effect, Astart consists of multiple mini-slow-start and mini-congestion-avoidance phases. Thus, cwnd does not increase as quickly as other methods, especially as cwnd approaches BDP.

This prevents the temporary queue from building up too fast, and thus, prevents a sender from overflowing a small buffer. In Astart, cwnd increase follows a smoother curve when it is close to BDP. In the case of a plurality of connections, each connection is able to estimate its share of bandwidth and switch to Congestion-avoidance at the appropriate time. In addition, Astart has a more appropriate (lower) slow-start exit cwnd, thanks to the continuous estimation mechanism, which reacts to the new traffic and determines an eligible sending rate that is no longer the entire bottleneck link capacity.

FIG. 1 is a block diagram of a computing device suitable for hosting a transport protocol control process in accordance with an exemplary embodiment of the present invention. A host 500 includes a processor 502 coupled via a bus 504 to a memory device 506, a storage device controller 508, and a network device controller 510. The processor uses the network device controller to control the operations of a network device 512 which is adapted for communications using a transport protocol to transmit data to a receiver 514 across a connection 516 through a computer network 518.

The storage controller is coupled to a storage device 520 having a computer readable storage medium for storage of program instructions 522 executable by the processor. The program instructions are stored in the storage device until the processor retrieves the program instructions and stores them in the memory. The processor then executes the program instructions stored in memory to implement the transport protocol control process as previously described.

Although this invention has been described in certain specific embodiments, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that this invention may be practiced otherwise than as specifically described. Thus, the present

embodiments of the invention should be considered in all respects as illustrative and not restrictive, the scope of the invention to be determined by claims supported by this application and the claims' equivalents rather than the foregoing description.

ABSTRACT OF THE DISCLOSURE

A modified TCP slow-start mechanism. When a connection initially begins or re-starts after a coarse timeout, modified TCP slow-start mechanism, called Adaptive Start (AStart), adaptively and repeatedly resets the slow-start threshold (ssthresh) based on an eligible sending rate estimation mechanism proposed in TCP Westwood. By adapting to network conditions during the startup phase, a sender is able to grow the congestion window (cwnd) quickly without incurring a risk of buffer overflow and multiple packet losses, thus improving link utilization under various bandwidth, buffer size and round-trip propagation times. The method avoids both under-utilization due to premature slow-start termination, as well as multiple packet losses due to initially setting ssthresh too high, or increasing cwnd too fast.

FLC/flc

MAS PAS496643.1-*-3/28/03 11:01 PM

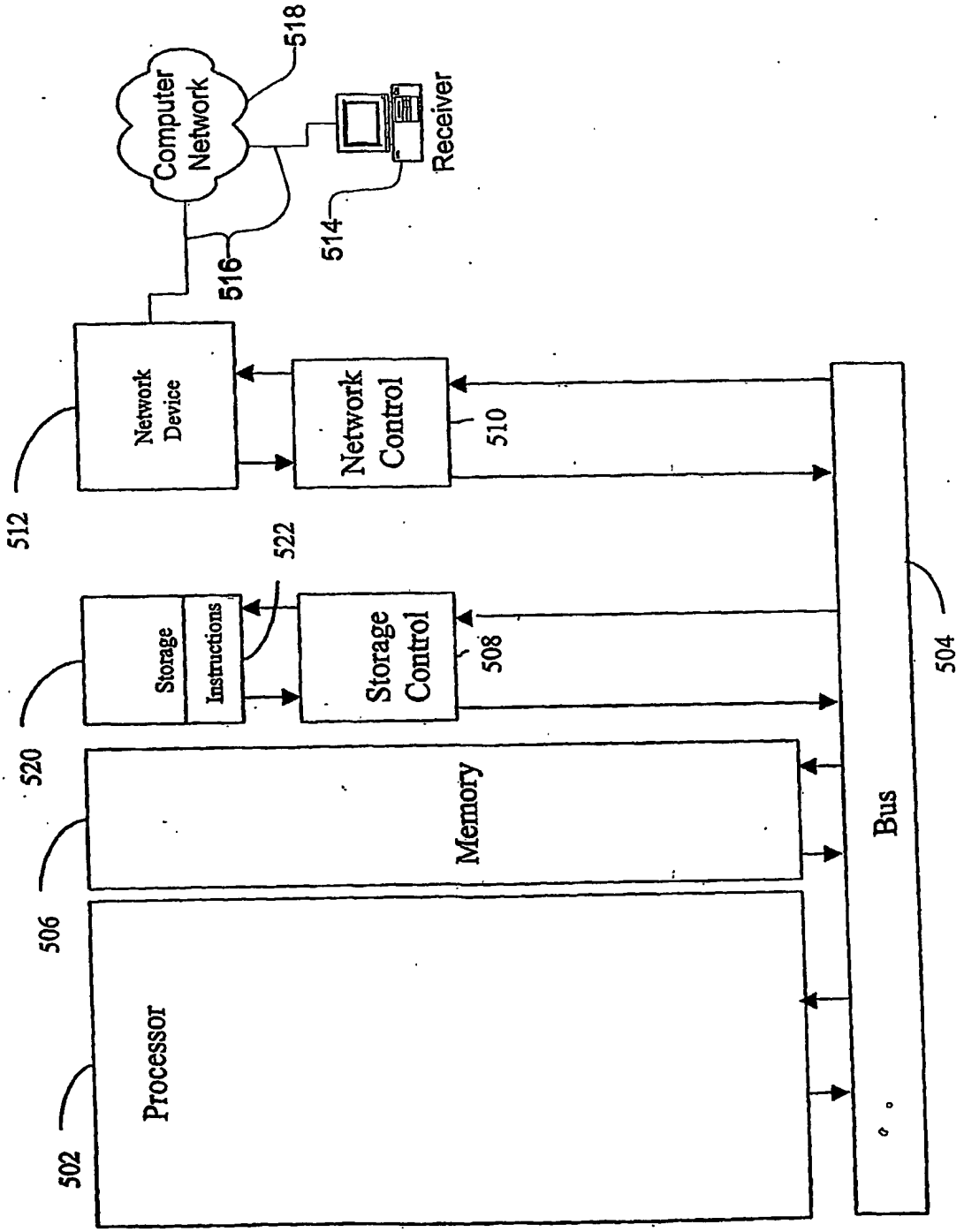
Computer Network
518
Receiver
514
516
Network Device
512
Network Control
510
522
Storage
Instructions
Storage Control
508
520
Memory
506
Processor
502
Bus
504

FIG. 1